#16
S.3rd
4/26/01

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re PATENT APPLICATION of

FARBER et al.

Group Art Unit: 2152

Examiner: Almari ROMERO

Appln. No. 09/612,598

Filed: July 7, 2000

For: **OPTIMIZED NETWORK RESOURCE LOCATION**

* * * * *

April 25, 2001

SUBMISSION OF EXHIBITS

Hon. Commissioner of Patents
and Trademarks
Washington, D.C. 20231

Sir:

Applicants submit herewith the four exhibits supporting the Rule 131 Declaration of Prior Invention (*DECLARATION OF ANDREW D. SWART UNDER 37 CFR § 1.131)*), filed April 20, 2001.

All of the pending claims being allowable, and all applicable requirements of 37 CFR § 1.607 having been complied with, it is respectfully requested that an interference be declared between the present application and with Leighton et al, U.S. Patent No. 6,108,703.

Respectfully submitted,
PILLSBURY WINTHROP, LLP

By _____
Brian Siritzky
Reg. No. 37497
Tel. No.: (202) 861-3702
Fax No.: (202) 822-0944

1100 New York Avenue, N.W.
Ninth Floor
Washington, D.C. 20005-3918
(202) 861-3000
30172416v1

Exhibit A Titled: Patent Claim: Network Reflector/Repeater

Patent Claim: Network Reflector/Repeater

Dave Farber
Sandpiper Software Consulting, L.L.C.
Draft 5 [redacted]

Contents

1. Abstract

In a networked computer environment in which a Client requests resources from a Data Service at an Origin Server, a *Reflector* process running on the Origin Server receives the requests and heuristically determines whether to handle the request locally or to *reflect* it by asking the Client

to submit the request to one of several *Repeater* processes running on Repeater Servers located elsewhere in the network. The determination of whether and where to reflect the request is based on analysis of the conditions at each Repeater relative to the requesting Client.

A request is reflected in one of the following ways:

- the Reflector returns a REDIRECT message identifying a selected Repeater in response to a Client request, causing the Client to direct its request to a different server.
- the Reflector assigns a *request ID* to the request, and, in response to the Client request, returns a REDIRECT message referring to the request ID at the selected Repeater. When the selected Repeater later receives a request for resource with the specified request ID, it obtains the corresponding resource from the Data Service at the Reflector. (This is expanded in section 7.4.3)
- the Reflector, when returning a resource to the Client, replaces references to the Origin Server embedded in the resource by references to the selected Repeater, so that subsequent requests using that reference are implicitly reflected.
- the Reflector, when returning a resource identifying a Data Service, replaces the resource with a new resource identifying a Repeater that supports a replicated version of the Data Service.

Before, during, and after the reflection process, additional communication and coordination can take place between the Reflector and the Repeater, as described herein.

Each Repeater handles a request through a Repeater Service. The Repeater Service may consist of a mirror, cache, or any other mechanism that can handle requests on behalf of the Origin Server. If the Repeater Service includes a cache, a missing cache entry is filled by forwarding the request to the Reflector, which in turn forwards the request to the Data Service.

2. Figures

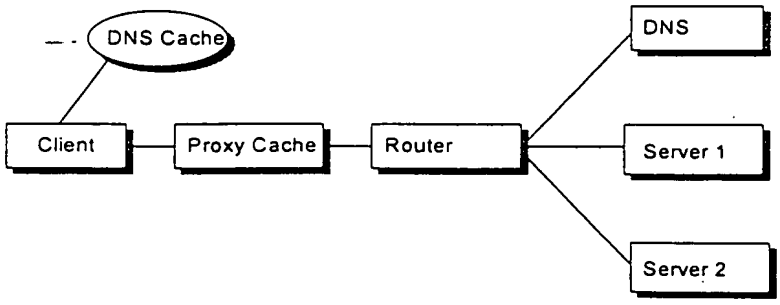

Figure 0.

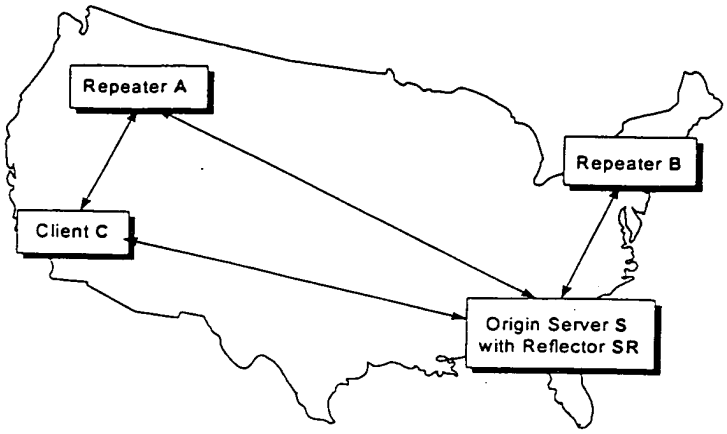

Figure 1

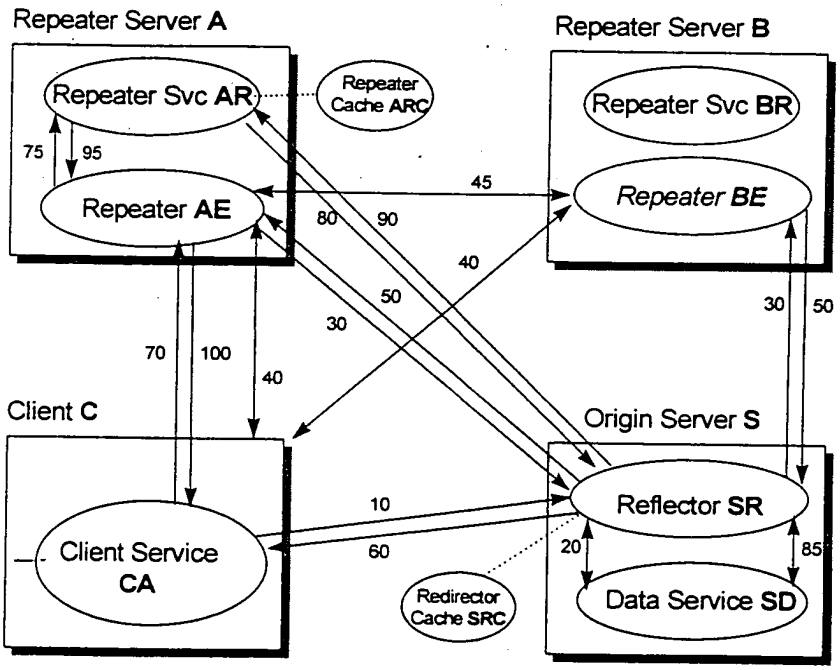

Figure 2.

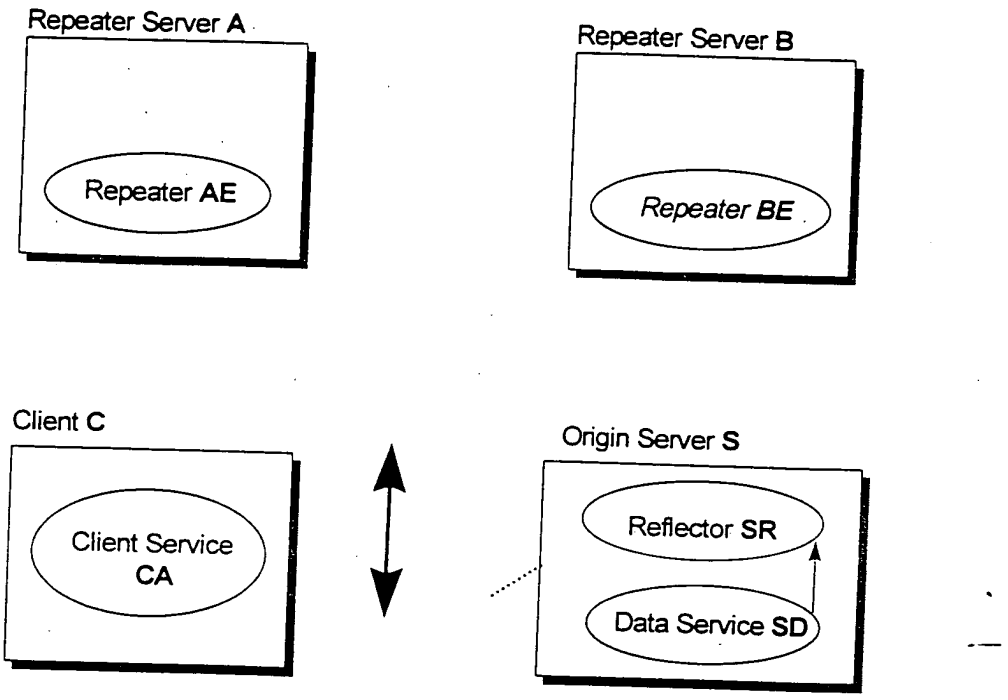

Figure 3.

3. Background of the Invention

Data traffic in the Internet and related wide-area computer networks is growing at a rapid pace. This growth is compounded by increasing demand for applications requiring high bandwidth, such as delivery of audio and video data, and is causing congestion in networks and servers, leading to poor or unavailable service. Therefore, methods and mechanisms are required to cope with this growth.

3.1.1. Existing Approaches

Beyond the "brute force" solutions of using faster computers and increasing network capacities, and "long term" solutions such as wholesale replacement of network protocols, substantial improvements can be made today by replicating resources in the network. Among the advantages of replicating resources are the following:

- The network load can be shared, decreasing the load on individual lines and servers
- Resources can be located nearer to the requester, reducing the distance data must travel and decreasing traffic sent over the network "backbone"
- The network as a whole may be more reliable, able to withstand failures of individual components

Assuming client protocols cannot easily change, there are two major problems in the deployment of replicated resources:

- What mechanism can cause a request for a resource, normally made to a single server, to be processed by an appropriately selected replicated copy of the server?
- What mechanism can allow the replication of selected resources (e.g., those in most demand) without requiring replication of all resources and services?

Several incomplete solutions have been proposed and partially implemented (see figure 0).

- Proxy caches may be used to handle requests on behalf of one or more servers. A proxy cache "impersonates" a server, and understands the protocol offered by that server. (The protocol of interest in the Internet today is called Hypertext Transfer Protocol, or HTTP.) If it has sufficient information to resolve a request, it does so; otherwise, it forwards the request to the "Origin Server". Proxy caches are widely used to reduce traffic on the network. Proxy caches only work well with static data (data which changes infrequently). Because the proxy cache must intercept all requests on behalf of a server, the proxy cache must be known to the client, or must take over the address of the server (in which case it is known as a "reverse proxy").
- DNS is sometimes used to select from among several identical servers. Distributed Naming Service (DNS) is used to by a client to resolve a server name to a server address. When identical replicated services are available, DNS can be used to select which server to use. Sometimes this is based on simple "rotation" to share the load. Recently, more intelligent techniques for selecting an address have been proposed. However, because DNS requests are made prior to requesting a specific resource, the replicated server must be capable of servicing any request. Furthermore, once the selection has been made, the client saves (caches) the server address, so the selection cannot easily be changed. It takes many hours for a change to DNS to propagate throughout the caches in a network.
- Routers may be modified to transmit conversations to one of several identical servers. A router connects several networks and manages traffic between them. A router uses the header on a message to determine its destination, and forwards the message to the appropriate network. Normally, messages must go to the identified system, because both sender and recipient maintain context that would otherwise be lost. A router could, by analyzing message headers, select among several identical servers to handle an individual session. Like the DNS solutions, a router solution requires that all replicated servers provide identical service. Furthermore, the router must be in the path between the requesting client and the selected server; this presents a serious problem if a number of replicated servers are deployed at distant locations.

Router and proxy cache solutions rely on *forwarding* requests to the appropriate server rather than *reflecting* them to the appropriate server. Such solutions cannot by their nature reduce traffic between the client and the router or proxy cache. The DNS mechanism described above is a kind of reflection, but is quite different from the form of reflection proposed here, as reflection must be performed for an entire system rather than selectively for individual resources in the system.

3.1.2. The Reflector/Repeater Invention

The invention described here provides an alternate, and better, way to ensure that requests are processed by the best replicated server. Like the solution employing only a proxy cache, the

invention works at the level of HTTP, as opposed to working at the routing or DNS layers of communication. Unlike the proxy cache solution, ultimate control remains at the Origin Server.

The invention does not intercept and process all requests, but in general causes the *client* to send requests to the best replicated server. In addition, the invention allows the origin server to coordinate handling of individual requests with the community of replicated servers, increasing its ability to respond to requests in a timely way.

4. Summary of the Invention

Consider a network including an HTTP *Client*, an HTTP *Origin Server*, and a number of replicated servers, which we will call *Repeater Servers* (see figure 1). Repeater servers are processors running software called *Repeater Services* that can be used to off-load requests submitted to the Origin Server. Typically, Repeater Services will either mirror selected resources provided by HTTP and FTP, or will serve as caches.

The invention is embodied primarily in two software modules: a *Reflector* module that runs on the Origin Server, and a *Repeater* module that runs on the Repeater Server. The Reflector causes requests for resources to be transparently reflected to the best Repeater Server. The Reflector communicates with the Repeater module on each Repeater Server to gather the information needed to make this decision.

The Reflector processes HTTP GET requests for resources at the Origin Server. The Reflector extracts from each request the address of the Client on which the request originated. It sends this origin address and other information to one or several Repeaters.

The selection of the best Repeater Server is a heuristic decision based on a number of approximate measurements and policy definitions. Each Repeater contacted estimates the approximate cost of communicating with the Client from the Repeater Server. The contacted Repeater may ask other Repeaters in a "Repeater Community" to provide it the same information. The cost estimate is based on a number of factors, including desired response time, server load, network transmission cost, and available network bandwidth. Each Repeater contacted returns its estimates in the form of a list identifying one or several selected Repeater Servers and the estimated cost components of using each.

The Reflector waits for a brief period, typically not more than a few seconds, for a response. It then selects the best Repeater Server from among all responses available. It uses the address of the selected Repeater Server to "reflect" the request, by creating an HTTP REDIRECT message containing the URL of the requested resource at the selected Repeater. It returns this message to the client. The REDIRECT message informs the Client that the requested resource is located on the selected Repeater.

When the Client receives the REDIRECT message, it extracts the new URL from the REDIRECT message, and then repeats the GET request, sending the new URL to the appropriate Repeater Server. This request is processed by the Repeater, which normally forwards the request to the Repeater Service. If the Repeater Service is a cache or mirror, and the requested resource is currently cached or mirrored locally, the Repeater retrieves the resource and returns it to the Client. If the resource is not cached, the Repeater requests the resource from the Origin Server.

The Reflector can reflect requests for resources even when the protocol for providing such resources does not include an explicit redirect capability. The Reflector does this by substituting Internet references that appear inside of resources delivered to the Client. Specifically, it

replaces URLs in HTML resources, and FQDNs or IP addresses inside MIME resources such as XDM resources.

Because of the cooperation between the Repeater and the Reflector, additional optimizations are practical. For example, when reflection is in process, the Reflector notifies the Repeater that a request for the indicated resource is coming, so that the Repeater can prepare to respond to it. Between reflections, the Reflector notifies the Repeater when it discovers that resources have changed; this helps to maintain cache consistency in the Repeater.

5. Glossary of Terms

This section defines a number of terms used in this document.

Note that some additional terms are defined at the beginning of section 6.

The following terms have been coined especially for the purposes of this document:

- Best repeater server - the most appropriate Repeater to service a client request, based on the cost estimates provided by each competing Repeater, taking into account network distance, network traffic, load, and service provided.
- Database resource - a resource which is identified or generated by a query process such as a search or transaction request.
- Forward - *v.* the process of sending a received request to another service on behalf of a client, allowing the service to process the request, and sending the services response back to the client. A proxy server forwards requests. Contrast with *reflect.*
- Network distance - the conceptual distance between two processors in a network, based on factors such as the number of network segments that must be traversed and the overall latency between them. Network distance is independent of geographic distance.
- Reflect - *v.* the process by which a server responds to a received request in a way that causes that request or future requests to be sent to a different server. (Contrast with *forward.*)
- Reflector - defined in section 4.
- Repeater - defined in section 4.
- Repeater Community - a group of cooperating Repeaters which provide services to a Client.

The following terms have commonly understood meanings within the industry:

- Absolute URL - a URL that indicates the Origin Server on which the requested resource resides, and the exact location of the resource on that Server.
- Cache - a temporary storage area which contains the results of recent requests for resources. When a resource is present in the cache and up-to-date, the server containing the resource does not have to request it from the origin server, saving time and reducing the demand on processors and networks.
- Cache validation - the process of confirming that a cached resource is up-to-date.

- Co-locate - to locate two computer systems nearby one another, generally on the same local area network, in order to improve their ability to communicate and/or to make it easier to administer them together.
- DNS - Domain Name Service, the system which converts FQDNs into IP addresses.
- FQDN - Fully Qualified Domain Name, the textual name of a Server or Client on the Internet or a similar network.
- FTP - File Transfer Protocol, a protocol that allows Clients to request files from Servers.
- HTML - HyperText Markup Language, a set of rules for describing the hypertext documents which comprise the vast part of the World Wide Web. HTML documents contain URLs (called "links") which refer to other resources.
- HTTP - HyperText Transfer Protocol, a protocol that allows a Client to request a resource from a Server.
- IP Address - Internet Protocol Address, the numerical address of a Client or Server on the Internet or a similar network.
- MIME (Multi-purpose Internet Mail Extensions) - Standards which define textual representation for non-textual objects such as images, sound, and references to servers. MIME notation is incorporated into HTML.
- Mirror - A replica of an Origin Server which provides all of the resources available on the Origin Server.
- Proxy server - a server which receives requests from clients and forwards them to servers. A "proxy cache" is a proxy server which caches responses from servers when possible.
- Relative URL - a portion of a URL which appears in a resource and which is interpreted relative to the address of the resource in which it appears (its "context"). The client is responsible for keeping track of the context and converting relative URLs to absolute URLs before making requests.
- Resource - data identified by a resource identifier such as a URL, and provided by a server in response to a request from a client.
- Reverse proxy - a proxy service which represents one or more Origin Servers and provides resources to clients on their behalf. A reverse proxy server usually includes a cache which eliminates the need to query the Origin Servers for most resources.
- Server - a computer system that provides resources requested by Clients.
- Static resource - a resource (for example, an HTML page or a file) which changes infrequently.
- Stream resource - a continuously generated resource, for example, a live audio or video feed.
- URL - Uniform Resource Locator, a textual name that identifies a resource (e.g. an HTML page) and the Origin Server on which it resides.

- Validator - a value associated with a resource and used to distinguish one version of the resource from another. A validator may be a serial number, a timestamp, or a hash code based on the current resource.

6. Brief Description of the Drawings

The invention consists of the following components (see figure 2):

- Client C– Processor requesting resources from Origin Server S.
- Client Service CA - the process making the request for resource through Client C on behalf of the end user. The Client Service is typically a user agent (e.g. a Web Browser) or a proxy for a user agent.
- Origin Server S – the server at which resources originate.
- Reflector SR – A process which intercepts requests from Client C and conditionally reflects them to a Repeater. The Reflector includes a cache SRC which maps addresses of Clients to best Repeater to use.
- Data Service SD – Any process or collection of processes that provide resources in response to requests from Client C.
- Repeater Servers A, B – Intermediate processors used to service client requests thereby improving performance and reducing costs in the manner described herein.
- Repeater AE and BE – A process which mediates requests on the Repeater Server and coordinates services with Reflectors. Among other things, the Repeater heuristically determines the cost to send messages between Client C and Repeater Services such as AR and BR.
- Repeater Service AR and BR – Any process or collection of processes that delivers resources to the Client on behalf of the Origin Server. In certain configurations, the Repeater Service will include a Repeater Cache ARC to avoid unnecessary transactions with the Origin Server.

The *Reflector* and *Repeaters* are unique components required for this invention.

Other components may be commonly available software programs; in particular, the invention is designed to work with any HTTP client (e.g. a Web browser), proxy, and server. However, to improve performance in specific environments, the Repeater Service and Data Service may be customized to take advantage of the cooperation between Redirector and Repeater, as described in various sections below.

In addition, the Reflector might be fully integrated into the Data Service (for instance, in a Web Server), and the Repeater might be fully integrated into a Repeater Service (for instance, a Proxy Cache). These components might be "loosely integrated" based on the use of extension mechanisms (such as so-called "add-in" modules) or "tightly integrated" by modifying the Service component specifically to support the Data or Repeater service.

7. Detailed Description of th Preferred Embodiment

An embodiment of the invention is now described in the following sections:

- 7.1 defines how requests are processed in general

- 7.2 details local processing of requests
- 7.3 discusses how the best Repeater is selected
- 7.4 describes how different Repeater Services service requests for different kinds of data
- 7.5 explains how Reference Substitution provides a type of reflection

This embodiment is specific to networks using HTTP protocols (and other protocols) communicated over TCP/IP. The description enumerates the processing steps required to provide a resource in response to a request identifying the resource by a URL (Uniform Resource Locator). See Figure 2 and Figure 3.

The exact order of steps given below may be changed in an implementation to permit the maximum processing overlap. The steps outlined below identify each of the interactions between processors required to service an entry. The use of caches and other optimizations make it possible to omit a number of these steps, as indicated below.

7.1. Processing Requests

7.1.1. Steps in Processing

10. The Client C sends an HTTP GET request containing the URL of a desired resource to the Origin Server.

12. The Reflector intercepts the GET request.

    In order to do so, the Reflector must be located on a system addressed by the Fully Qualified Domain Name (FQDN) embedded in the URL, and it must use the port that would otherwise be assigned to the Data Service's HTTP service.

    The Reflector need not be located on the same processor as the Data Service, but it is assumed to be at least co-located, for purposes of management and for access to information described in step 20.

14. The Reflector analyzes the request, confirms that the URL is valid, and determines whether the URL should be reflected. The rules for this determination will be based on configuration data. For instance, it may be inappropriate to reflect a URL if:

    - the URL identifies a small static resource with no internal links
    - the URL indicates that custom processing is necessary (for instance, a query URL containing a "?" character)
    - the URL identifies a resource which is not cacheable because it continuously changes
    - reflection is currently disabled by the Origin Server operator as a matter of policy
    - the request was sent by a Repeater

    (Note that the ability to inhibit reflection selectively is a characteristic that distinguishes this invention from DNS and Router approaches to this problem.)

    If reflection is inhibited for a given request, go to step 200.

18. If Repeaters provide cache services, and if the request is a *Modular Query* (defined in section 7.4.3 below), the Reflector creates a Request ID to identify the result later.

20. If Repeaters provide cache services which can use validators and the request identifies static data, the reflector retrieves a validator from the Data Service. The validator will be used to identify the current version of the requested resource.

    To improve performance in some configurations, the Reflector and Data Service may communicate over a persistent, low latency connection.

25. The Reflector finds the best Repeater Server to process the request. The details of this process are described in steps 28-53, presented in section 7.3 below.

55. Processing continues after the best Repeater Server has been selected; in figure 2, it is assumed that the best Repeater Server is A.

    Given this selection, the Reflector determines whether it should forward the request to the local Data Service or reflect it to the best Repeater. If the request should be forwarded, go to step 200.

    Otherwise, continue in step 58.

58. The Reflector composes an HTTP REDIRECT message. The REDIRECT message includes several components:

    - the FQDN of the best Repeater Server, as selected above in step 25
    - the FQDN of the Origin Server which was the original source of the request
    - optionally, a request ID and/or validator as determined above in step 20

60. The Reflector returns the REDIRECT message to the Client.

65. If the selected Repeater Service includes a cache, the Reflector notifies the selected Repeater of the request. This allows the Repeater Service to pre-load the cache, in case the indicated resource is not already in the cache.

70. The Client resubmits the original GET request to the identified Repeater, including all modifications to the URL provided by the Reflector.

72. The Repeater receives the GET request.

73. The Repeater analyzes the URL to determine the identity of the Origin Server from which the request was reflected.

74. The Repeater checks that the request identifies a URL that the Repeater is prepared to service. This may not be the case for several reasons, for example:

    - the Repeater has become too busy to service the request
    - the Repeater no longer supports the identified Origin Server
    - reflection has been disabled by the Origin Server operator as a matter of policy
    - the URL identifies a processing task that cannot be handled by the Repeater
    - a redirected URL has expired. This may happen if the URL was kept for too long by the client -- for instance, in a cache or a bookmark.

In such cases, the Repeater builds a REDIRECT message to the Data Service, and returns it to the Client Agent. Care is taken to ensure the new REDIRECT response does not lead to an infinite loop. Continue with step 10.

76. The Repeater requests the associated Repeater Service to locally retrieve the resource indicated by the URL. The processing performed depends on the Repeater Service provided. Details are discussed in section 7.4 below.

If the resource is available to the Repeater, skip to step 96. Steps 80-94 specifically describe the processing when the Repeater Service includes a cache and the cache does not contain the correct version of the resource.

80. The Repeater Service forwards the requested URL to the Reflector.

In the preferred embodiment, the Repeater Service processes a cache miss by forwarding the request to the Reflector. For better performance, but with some concomitant loss of flexibility, it could alternatively forward the request directly to the Data Service.

82. If the request includes a Request ID:

- the Reflector waits for the processing of the Modular Query started in step 198, corresponding to the Request ID.
- The Reflector creates a resource from the result of the Modular Query.

  The Reflector converts references to database records in the result of the query into relative URLs as part of the response. This will cause the Client to request records in the result from the Repeater, where they may be cached.
- Go to step 90.

85. The Reflector forwards the request to the Data Service, which fulfills the request.

90. The Reflector returns the requested resource to the Repeater Service.

92. The Repeater performs *Reference substitution*, which effectively redirects references internal to the resource. Reference substitution makes it possible to "proactively" reflect resource requests, even when the protocols for such resources do not support redirection, such as FTP. Reference substitution is described in detail in section 7.5 below.

94. The Repeater Service caches the resource received from the Reflector, if appropriate.

96. The Repeater replaces embedded references to resources by the resources to which they refer. Such embedded references may be the results of Modular Queries.

Providing this form of expansion at the Repeater site eliminates the need to redundantly transfer common units of data ("boilerplate") to the Repeater. This capability is analogous to a manufacturing and distribution system in which it is more economical for certain final assembly steps to be performed by the distributor than by a central manufacturing facility.

100. The Repeater Service returns the resource to requesting Client. The process of servicing a request is now complete.

7.2. Local Request Processing

If the Reflector determines that a request should be handled locally, the following steps are taken. Either the request is for a stream object and should be reflected, or the request is for a local resource and should be forwarded to the Data Service at the same Origin Server (or co-located with it). See Figure 3. These steps are taken if the Reflector determines that reflection should be inhibited (step 14) or that the Origin Server is effectively the best Repeater Server to use (step 55).

7.2.1. Steps in Processing

200. The Reflector forwards the request received directly to the Data Service.

205. The Data Service fulfills the request (as in step 85).

210. The Data Service returns the resource to the Reflector.

220. The Reflector performs *reference substitution* on the resource, as described in section 7.5 below.

299. This ends Reflector forwarding.

7.3. Selecting the Best Repeater

This section describes step 25 and, equivalently, step 230.

The best Repeater Server to service a given request is identified in the following steps:

1. the Reflector sends a copy of the Client request to one or more Repeaters to obtain *estimates*,
2. each contacted Repeater creates an estimate which identifies the cost of sending resources from the various Repeaters in its community to the requesting Client, in terms of several metrics;
3. the contacted Repeaters return estimates to the Reflector.
4. the Reflector compares these costs and from them selects the best Repeater Server.

This process is analogous to a bidding system. The Reflector creates a Request for Quotation and sends it to one or more Repeaters. Each Repeater bids for the contract. The Reflector awards the contract to the Repeater with the lowest bid.

It is not required that the Repeater AE and Repeater service AR run on the same processor. However, they must at least be co-located for some of the cost estimation methods to work correctly.

In this invention, individual Repeaters may be managed by competing service companies. Some means is necessary to assure that the Repeaters give estimates "in good faith" and that they are accountable for their estimates. Such means are not addressed by this invention.

7.3.1. Steps in Processing

The following steps explain the process in more detail.

28. The Reflector determines which Repeaters to contact regarding this transaction.

If there is only one Repeater, or if the Reflector Cache SRC already contains a Best Repeater Address for the given Client and service type, the Best Repeater Address has been determined, and no further steps are taken.

Repeaters may be grouped into Repeater Communities, consisting of a set of member Repeaters. Thus, the Reflector is able to ask a single Repeater for estimates that cover the community. The contacted Repeater then represents the community and creates estimates for its members.

The Reflector Cache SRC contains estimates for individual Repeaters. The Reflector need only perform steps 30-52 for those Repeaters for which it does not have an up to date estimate of costs.

The Reflector Cache SRC is made more efficient by noting that several Clients have effectively the same address. For instance, if two Client addresses are members of the same "autonomous system" then the same estimate can be provided for them by a Repeater.

30. The Reflector sends information about the bid to each selected Repeater (in figure 2, Repeaters AE and BE). Such information may include some or all of the following

    - the requested resource URL
    - the type of service (based on the URL)
    - the address of the requesting Client
    - the service class of the requesting Client

    In certain configurations, the Reflector may communicate with a Repeater over a persistent connection to eliminate the connection overhead between these processes. For added security at the cost of additional processing, messages sent over this channel may be authenticated and encrypted, to prevent the use of Repeaters by unauthorized Reflectors, and to keep the information exchanged private.

    For improved performance at a cost of additional complexity, a Repeater may broadcast the request to other Repeaters in the same Repeater Community, and aggregate the responses.

40. Each contacted Repeater estimates the cost in terms of several metrics, as described in section 7.3.2 below.

50. Each contacted Repeater returns its estimate to the Reflector or Repeater that requested it.

51. The Reflector waits to receive all estimates, or until a brief waiting period elapses, whichever occurs first. Repeaters which are not assigned estimates by this procedure are eliminated from the selection process (essentially by assigning an infinite cost to each). If no estimates are received within the waiting period, go to step 53.

52. The Reflector takes all available estimates and reduces each one to an individual overall cost, according to the weighting procedure described in section 7.3.3 below.

53. The Reflector selects the Repeater with the lowest overall cost. The Reflector may at this point conclude that the Origin Server on which the Reflector is running can service the request at a lower cost than any Repeater.

7.3.2. Estimating the Cost for One Repeater

This section describes step 40 in more detail.

Cost is determined using dynamic measurements in real time, and/or using static calculations -- that is, calculations based on information that is currently available, changes infrequently, and is assumed to be correct. In some cases, heuristic techniques are used to measure components of the cost.

To allow for tradeoffs between accuracy of calculation, speed of calculation, and stability of results, the operator or network manager can select which components should be provided in the cost calculation.

Each Repeater is responsible for determining the cost between itself and a requesting Client. However, when a calculation does not require dynamic measurements, a single Repeater can calculate the cost for any or all Repeaters in its Repeater Community.

Two kinds of cost measurement can be distinguished: network costs, and other costs.

To compute some network costs, the Repeater first converts the Client IP address to an Autonomous System (AS) number, according to well-known means. The AS number represents a family of IP addresses with essentially the same route and cost. Network cost estimates include one or more of the following metrics:

- *static cost based on selected direction* - a static indication of the path taken away from the Repeater is determined a follows. The Repeater sends a TCP/IP ICMP messages to the Client, with a very small time-to-live (TTL) value. This results in an ICMP response from one of a small number of intermediate systems within the "radius" defined by the TTL. The identity of the intermediate system is associated with a cost assigned to that path.

  As the TTL increases, the number of intermediate systems increases. The Repeater operator is responsible for maintaining the table assigning costs to each intermediate system.

- *cost based on destination AS* - a statically derived cost is determined by first converting the Client IP address to an Autonomous System (AS) number, according to well-known means, and then looking up the cost from the Repeater to the Client AS. (The AS number represents a family of IP addresses with essentially the same route and cost.)

  The table is statically created, and updated regularly, using available Internet routing information. It shows the path between the Repeater and each AS. Each path consists of a series of links between Autonomous Systems in the path. Costs for selected links are assigned manually by the operators of the Repeater Community; or derived from available information about each AS. The final cost is the sum of costs over each link. Because routing data changes constantly, the table is seldom exactly correct. Nevertheless, as a heuristic it is quite effective.

  [*Note:* If it seems appropriate, I can show a description of this implementation. But it has not yet been determined, will be relatively complex, is probably described in the literature somewhere, so I'm not sure it will add value.]

- *dynamic number of hops* - an dynamically derived indication of the number of systems forwarding messages between the Repeater and the Client is measured by using TCP/IP ICMP messages with a range of TTL settings, in a way analagous to the way the

"traceroute" program identifies hops taken by a message traversing the path from Repeater to Client. Because not all networks respond to ICMP messages, this measurement is not always fully available or accurate, but when it is, it can be used to provide a rough estimate of the relative latency and bandwidth between a client and server.

- *time to respond* - a dynamic indication of the time required to send a message between the Client and the Repeater. The time to respond may be measured by calculating the round trip time of an ICMP message sent to the Client. Because not all networks respond to ICMP messages, this measurement is not always available or accurate, but when it is, it provides a good estimate of the current latency between the Repeater and Client.

In addition to network cost estimates, other information is important in selecting the best repeater:

- *load* - a measure of how busy the Repeater is. It may be preferable, for instance, to send a request through an expensive network connection to an available server rather than through an inexpensive network connection to a busy server. Most servers have well defined metrics available for representing the load numerically.
- *service type* - an indication of whether the Repeater supports a given type of service. This indication is statically configured for each Repeater. For example, some Repeaters may service HTTP requests, but not FTP or Audio requests. (Service type is not really a "cost", indicates whether a given Repeater is eligible to service a request at all.)

The cost values produced by the Repeater are returned to the requesting Reflector or Repeater. If a single Repeater is aggregating estimates for a number Repeaters in a Repeater Community, it returns its response when the best estimates are known and before the delay time has expired.

An individual Repeater might not complete its measurement in time for the Reflector to take advantage of it for a given request. However, if the measurement arrives later at the Reflector, the Reflector will cache it for later use.

7.3.3. Comparing Costs

This section describes step 54 in more detail.

The Reflector uses the multi-dimensional results of the Repeaters' estimation process described above to select the best Repeater Server. The technique for comparing these values is configurable by the Reflector operator.

Different Reflectors may request different levels of service. In this case, each level of service would correspond to a different configuration.

A simple, linear comparison mechanism is as follows. A set of coefficients is assigned, one coefficient for each element of the estimate metric. The specific coefficients are configurable, and depend on the level of service desired and the service type requested. Each estimate is reduced to a scalar overall cost by multiplying the components (Mi) by their respective coefficients (Ci) and summing the results:

Sum over i (Mi * Ci)

The repeater with the lowest overall cost is selected as the best repeater.

In the preferred embodiment of the invention, instead of linear coefficients, thresholds are used for each measurement. Upper and lower thresholds (Ai and Bi, respectively) are defined for each measurement, along with costs to assign when measurements are outside these limits (Xi and Yi, respectively). Thus,

Sum over i (if Mi > Ai then Xi else if Mi < Bi then Yi else Mi * Ci)

7.4. Data Service and Repeater Service

A Data Service, as defined above, may use one or more arbitrary strategies for retrieving resources given requested URLs. The most common Data Services provide static resources -- that is, given a URL, they generally provides the same resource, though the resource may be changed occasionally. The type of Repeater Service that can be used will depend on the strategy used by a Data Service.

The following table lists three common Data Service strategies, and shows the corresponding Repeater strategy required to take advantage of it. Each strategy is then discussed in more detail.

Data Service	**Repeater Service**
Static resources	Cache or mirror.
Stream resources (Live audio/video)	Multiplex data stream received using multicast or dedicated insertion point.
Static database resources	Cache or mirror. To cache, Origin Server must handle URLs representing Modular Queries and requesting individual records.
Dynamic database resources	Mirror or unsupported. An Origin Server that offers static resources with relatively few database resources can service the database resources locally, and reflect only requests for static resources.

7.4.1. Static Resources

Static resources are resources that change infrequently. The Repeater Service provides access to these resources when configured either as a mirror (wherein an exact copy of all resources is provided by the Data Service), or a cache (wherein resources are copied from the Data Service on demand and retained for a limited period of time).

Software is commonly available to permit the creation of both mirror servers and of HTTP caches, and is not discussed further herein.

7.4.2. Stream Resources (live audio/video)

Stream resource requests result in a high bandwidth data stream that is typically generated in real time (e.g., a radio broadcast). The Repeater Service offers replicated access to this data stream by multiplexing an audio/video data stream received from a more central source (e.g., an origin server).

A given Origin Server can only offer a limited number of multiplexed output channels to requesting clients. The Reflector/Repeater invention increases the number of servers that can handle requests, and thus multiplies the number of available channels, without requiring changes to routers, client protocols or the user interface.

(Multicasting also addresses this problem, but it requires routing mechanisms that are not widely available.)

Software for multiplexing a single input into multiple outputs is available today and is not the subject of this patent.

7.4.3. Database Resources

Database requests involve a query or search through many records to find those that meet a user's requirements. A Repeater Service can provide services by mirroring the entire database. Alternatively, the Data Service can support *modular queries.*

A modular query is defined as a Query URL that results in an HTML containing references to static data objects which are records in the database. The Data Service must support URL-based requests for both queries and component data objects.

For example, consider a database containing descriptions of products. The corresponding Data Service supports modular queries if it accepts URLs requesting information about particular products, where each such request URL generates an HTML page containing references to resources describing the selected products. The embedded references are themselves URLs identifying static records; they might identify HTML pages that contain photographs and textual descriptions of the products.

Because the result of a modular query is returned by the Repeater Service, modular queries are used in conjunction with *request IDs.* Request IDs (described in steps 18, 58, and 82 above) allow the Data Service to query the database while a reflection is in progress, as follows. When a database query request is received, a request ID is assigned to the request, and the processing of the request begins. A Repeater to service the request is selected, and a URL including Repeater name and the request ID is created and reflected to the Client. The Client receives the REDIRECT request and sends the request to the Repeater. When the Repeater receives the reflected request from the Client, it then obtains the result of the query from the Reflector.

7.4.4. Steps in Processing

This section describes step 76 in more detail.

- If the selected resource is mirrored, the Repeater's local copy of the resource is returned to the client.
- If the Repeater Service provides a cache, the resource is retrieved from the cache using standard cache processing techniques. If the resource is available in the cache and known to be up-to-date, it is returned. If it is available in the cache but may not be up-to-date, the Repeater Service can check with the Data Service to see whether the cache should be invalidated. If the resource in the cache is invalid, the Repeater Service attempts to replace the resource by the latest resource from the Data Service. If the resource is not available from the Data Service, the Repeater Service returns the old resource from the cache, if any, or the appropriate error code.

  In addition to standard caching techniques, if the requested URL includes a validator, the caching algorithm can use it to determine whether the requested version of the resource in the cache is the same as the version resource at the Origin Server.

- If the selected resource is a stream, the Repeater must establish access to the stream identified and provide a copy of the stream to the client. It may need to contact the Origin Server or some other source to gain access to the stream.

7.5. Reference Substitution

This section describes steps 92 and 220 in more detail.

Because the HTTP REDIRECT message can only be used to reflect HTTP resource requests, an alternate mechanism is used to reflect requests for other protocols, such as the File Transfer Protocol (FTP), or the PNM protocol used by Progressive Networks' RealAudio. This mechanism is referred to as *reference substitution*.

The reference substitution mechanism scans a given resource containing embedded references and replaces the specific references found in it by references to the corresponding resource at the selected Repeater. Specifically, this mechanism replaces URLs found in HTML resources, and embedded service addresses in MIME resources.

For resource requests that cannot be reflected, reference substitution must be performed at the Reflector (in step 220 above). In the preferred embodiment, reference substitution for reflected requests is performed by the Repeater (in step 92 above), on the assumption that the Repeater has more available processing power to perform such a task.

7.5.1. HTML

URLs that occur in HTML pages are replaced by URLs identifying the selected Repeater Server. The new URLs may be relative URLs, which implicitly refer to the server providing the HTML page in which they are embedded. They may also be absolute URLs, which refer to a specific server.

Depending on configuration parameters, the reference substitution mechanism considers only a subset of all URLs embedded in an HTML page to be "eligible" for reference substitution. The eligible URLs may be relative URLs, absolute URLs which refer to the Origin Server, or URLs indicating a specific protocol such as FTP.

The reference substitution mechanism for URLs works in the following steps:

- For each eligible URL in the HTML page
  - Determine the best Repeater Server for the indicated resource (as defined above in section 7.3)
  - Replace the URL by a URL identifying the same resource on the selected Repeater Server

7.5.2. MIME

Some MIME resources contain references to servers that provide streaming audio and video, in the form of IP addresses or FQDNs. For instance, the x-xdma MIME type used in Xing Technology Corporation's StreamWorks contains a reference to a StreamWorks audio server.

The reference substitution mechanism for embedded services in MIME objects works in the following steps:

- Confirm that the MIME resource contains an embedded reference
- Confirm that the reference embedded in the MIME resource identifies a known Origin Server
- Select the best Repeater Server for the Origin Server to handle the request
- Replace the embedded reference by a reference to the same resource on the selected Repeater Server.

8. What We Claim

What is *not* unique:

- intercepting a request at a server
- extracting its origin
- creating a REDIRECT message and returning it to the requester
- forwarding a request to another server
- individual notions for measuring cost or distance

What *is* unique:

- the use of reflection as a way to off-load demand from a server to a selected Repeater
- the use by the reflector of estimates created by repeaters as a way to select the best Repeater from among several possible choices (steps 28-52)
- the location of estimation mechanisms at each Repeater (steps 28-52)
- the use of Reference Substitution as a means of reflection (step 92)
- the use of simultaneous reflection and computation to speed query processing, through Request IDs (step 20)
- assembling a completed resource at a Repeater before returning it to the Client (step 96)
- the use of the above techniques specifically for HTTP and MIME resources

-- end --